2.13 TOPOGRAPHIC CHARACTERISTICS

Topographic effects in mission-level models are effects due to terrain or due to other natural or man-made features. Man-made features include buildings, bridges, machinery, etc., while natural features include forests, rivers, swamps, etc. Although topographic features could affect many aspects of a scenario (e.g., movement or decision-making), the effects discussed in this section are their effects on energy transmission; i.e., clutter, multipath, diffraction, and line of sight masking. SWEG does not model clutter, multipath, or diffraction, but does it simulate masking effects for sensing, communicating, and jamming.

SWEG uses processed Defense Mapping Agency (DMA) digital terrain elevation data (DTED) to determine terrain masking effects. Any of the standard DMA terrain data base formats can be read and processed by the GDB step and then further processed by the EDB step. If the DMA terrain ‘squares’ used are at different longitudinal densities, all of them must be processed at the lowest density (measured in seconds of arc) in the scenario. (This would apply, for example, if the scenario included terrain both above and below 50 degrees latitude.)

The GDB step converts the terrain data from points to (flat) triangular areas that adjoin one another to produce a continuous but non-differentiable surface. All triangles used are approximately equilateral with sides of length equal to 2n times the distance between DMA points. By selecting a minimum and a maximum value of n (greater than or equal to one), the user may select triangles of different sizes to account for varying terrain smoothness; i.e., SWEG may be instructed to create smaller triangles whenever the first triangle created gives an altitude at a DMA latitude and longitude that is unacceptably different (from the user’s perspective) from the DMA altitude there. The smallest triangle available has sides with length equal to two times the distance between DMA points.

Masking Effects

Masking is an interruption in energy transmission due to physical obstruction in the line of sight between an energy transmitter and an energy receiver. A physical obstruction can be natural or man-made or it can be simply the curvature of the earth. In mission-level models, masking may interfere with the operation of communication devices, sensors, and jammers.

If the user provides the appropriate instructions, SWEG can model masking due to earth curvature, to terrain, and to other features. Masking can interfere with the operation of sensors, communication devices, and jammers in SWEG.

2.13.1 Functional Element Design Requirements

This section contains the design requirements necessary to implement masking in SWEG 6.5.

a. SWEG will model masking due to the curvature of the earth whenever the user supplies an effective earth radius. When specified, masking due to earth curvature can affect energy transmission for sensing, communicating, and jamming.

b. SWEG will model masking due to terrain whenever the user supplies DMA terrain data. When terrain is defined, terrain masking can affect sensing, communicating, and jamming.

c. SWEG will model masking due to other natural and man-made features when so instructed by the user. The user may define the shape of feature types and the types of energy receivers that can be masked by each. Specific features of any defined type may be placed at arbitrary locations in the scenario. When applicable, masking by features will affect sensing, communicating, and jamming.

2.13.2 Functional Element Design Approach

Before calculating signal level for sensing, communicating, or jamming, SWEG checks the line of sight between the transmitter and the receiver for masking due to terrain, cultural or natural features, or the curvature of the earth.

Terrain is modeled as described in Section 2.16. For masking purposes, cultural or natural features can be represented in SWEG as platforms of a player. In the TDB types of shapes can be defined that are linear (with or without a vertical extent), or shapes that have volume or area. Shape types with volume or other vertical extent can be defined as opaque to specific types of receivers. Also in the TDB, types of platforms can be given shape types. In the SDB, specific instances of the platforms with shapes can be given individual dimensions as well as locations.

2.13.3 Functional Element Software Design

This section contains two tables and four software code trees which describe the software design necessary to implement the requirements and design approach outlined above. Tables 2.13-1 and 2.13-2 list most of the functions found in the code trees, and a description of each function is provided. Figure 2.13-1 depicts the path from main to loschk, the top-level C++ function within the code for masking due to earth curvature or terrain. Figure 2.13-2 contains the code tree for loschk and its subordinate functions. Figure 2.13-3 depicts the path from main to featlos, the top-level C++ function within the code for masking due to other natural or cultural features. Figure 2.13-4 contains the code tree for loschk and its subordinate functions.

A function’s subtree is provided within the figure only the first time that the function is called. Not all functions shown in the figures are included in the table. The omitted entries are trivial lookup functions (single assignment statements), list-processing or memory allocation functions, or C++ class functions for construction, etc.

TABLE 2.13-1. loschk Functions.

Function	Description
BaseHost:Run	runs all steps
BSRVevent	controls sensor physical processing
BSRVmasking	performs masking calculations for sensor systems
BSRVonechance	supervises sensor chance calculations
loschk	checks line of sight between two objects
main	controls overall execution
MainInit	initiates processing and runs either the boot step or normal execution
MainParse	controls parsing of user instructions
program	controls overall execution of all steps except bootstrap
semant	controls semantic processing instructions
simnxt	controls event sequencing and runtime execution
simphy	controls processing of physical events
simul8	controls semantic processing of runtime instructions
TTerrain::EdgeMasklos	determines the masking of terrain edges
TTerrain::FindTriangle	determines the terrain triangle for a point given the x, y coordinates
TTerrain::LineOfSight	determines if there is a line of sight between two objects
yakker	determines next communications event for a given net
yaksig	determines signal level at receiver

```
main
   |- BaseHost:Run
      |- MainInit
         |- program
            |- MainParse
               |- semant
                  |- simul8
                     |- simnxt
                        |- simphy
                           |- BSRVevent
                           |  |- BSRVonechance
                           |     |- BSRVmasking
                           |        |- loschk
                           |- yakker
                              |- yaksig
                                 |- loschk
```

FIGURE 2.13-1. loschk Top-Level Code Tree.

```
loschk
    |-TMaster::GetTerrain
    |-TMaster::TerrainOn
    |-DVector::Getz
    |-dbg_acos
    |   |-MathExcpt::MathExcpt
    |   |   \-SwegExcpt::SwegExcpt
    |   |       \-StringDup
    |   \-acos_c
    |-operator-
    |   |-DVector::DVector
    |   |-DVector::Getx
    |   |-DVector::Gety
    |   \-DVector::Getz
    |-DVector::GetHorizLength
    |   \-dist
    |-DVector::Getx
    |-DVector::Gety
    \-TTerrain::LineOfSight
        |-DVector::Getz
        |-DVector::Getx
        |-DVector::Gety
        |-dist
        |-dbg_acos
        |-VertexIndex::VertexIndex
        |   \-VertexIndex::operator=
        |-TTerrain::FindTriangle
        |   |-VertexIndex::VertexIndex
        |   |-TAddress::Cartesian2Spherical
        |   |   |-dbg_sqrt
        |   |   |   \-MathExcpt::MathExcpt
        |   |   |-dbg_asin
        |   |   |   |-MathExcpt::MathExcpt
        |   |   |   \-asin_c
        |   |   \-dbg_atan2
        |   |       \-MathExcpt::MathExcpt
        |   |           \-SwegExcpt::SwegExcpt
        |   |-TMessages::WriteMessage
        |   |   |-TMessages::GetMsg
        |   |   |-TMessages::PrintALine
        |   |   |   \-TSeqFile::Write
        |   |   |       \-MFiles::Append
        |   |   \-sysdun
        |   |       |-TActWindow::GetNext
        |   |       |-TActWindow::~TActWindow
        |   |       \-ProgramStop::ProgramStop
        |   |           \-SwegExcpt::SwegExcpt
```

FIGURE 2.13-2. loschk Code Tree.

```
|    |-TAddress::Spherical2Cartesian
|    |    \-TMaster::DebugOn
|    |-VertexIndex::operator=
|    |-VerticeArray::operator[]
|    |    \-VertexIndex::Value
|    |-TTerrain::toPtr
|    |    \-SwegExcpt::SwegExcpt
|    |-operator-
|    |    \-VertexIndex::VertexIndex
|    |-VertexIndex::operator++
|    |-dist
|    |-operator+
|    |    \-VertexIndex::VertexIndex
|    \-VertexIndex::operator+=
|         \-operator+
\-TTerrain::EdgeMasklos
     |-dist
     |-VerticeArray::operator[]
     |-operator+
     |-VertexIndex::operator+=
     |-isZeroEquiv
     \-TTerrain::toIndex
          \-SwegExcpt::SwegExcpt
```

FIGURE 2.13-2. loschk Code Tree. (Contd.)

TABLE 2.13-2. featlos Functions.

Function	Description
BaseHost:Run	runs all steps
BSRVevent	controls sensor physical processing
BSRVmasking	performs masking calculations for sensor systems
BSRVonechance	supervises sensor chance calculations
featlos	determines if shape interferes with line of sight
main	controls overall execution
MainInit	initiates processing and runs either the boot step or normal execution
MainParse	controls parsing of user instructions
numerical	sorts address codes
program	controls overall execution of all steps except bootstrap
region	determines if a point is within a two dimensional region
semant	controls semantic processing instructions
simnxt	controls event sequencing and runtime execution
simphy	controls processing of physical events
simul8	controls semantic processing of runtime instructions
TAddrData::GetShapeList	finds the shape list at a given address

TABLE 2.13-2. featlos Functions. (Contd.)

Function	Description
TAddress::GetAddresses	retrieves a sorted collection of addresses between two points
TAddress::GetCellRadius	determines the radius of an address tree cell
TAddress::GetShapeList	creates a list of shapes that might affect line of sight
TAddress::InsertVertCodes	inserts address codes, including parents, for a given point
TAddrNode::GetNode	returns a pointer to the address data added or found
TMemory::Allocate	allocates permanent storage
TMemory::AllocTemp	allocates temporary storage
TMemory::Deallocate	deallocates a list of blocks by using the address within the provided pointer
TMemory::DeallocFront	deallocates storage
TMemory::DoAllocate	allocates either permanent or temporary storage
TTable::SearchInt	searches a table for a specific integer
TTerrain::Elevation	determines the z-coordinate on a surface given the x, y coordinates
TTerrain::FindTriangle	determines the terrain triangle for a point given the x, y coordinates
yakker	determines next communications event for a given net
yaksig	determines signal level at receiver

```
main
    |- BaseHost:Run
        |- MainInit
            |- program
                |- MainParse
                    |- semant
                        |- simul8
                            |- simnxt
                                |- simphy
                                    |- BSRVevent
                                    |   |- BSRVonechance
                                    |       |- BSRVmasking
                                    |           |- featlos
                                    |- yakker
                                        |- yaksig
                                            |- featlos
```

FIGURE 2.13-3. featlos Top-Level Code Tree.

```
featlos
    |-TMaster::GetTerrain
    |-TMaster::TerrainOn
    |-DVector::Getx
    |-DVector::Gety
    |-DVector::Getz
    |-TTerrain::Elevation
    |    |-DVector::Getx
    |    |-DVector::Gety
    |    |-VertexIndex::VertexIndex
    |    |    \-VertexIndex::operator=
    |    |-TTerrain::FindTriangle
    |    |    |-VertexIndex::VertexIndex
    |    |    |-TAddress::Cartesian2Spherical
    |    |    |    |-dbg_sqrt
    |    |    |    |    \-MathExcpt::MathExcpt
    |    |    |    |         \-SwegExcpt::SwegExcpt
    |    |    |    |              \-StringDup
    |    |    |    |-dbg_asin
    |    |    |    |    |-MathExcpt::MathExcpt
    |    |    |    |    \-asin_c
    |    |    |    \-dbg_atan2
    |    |    |         \-MathExcpt::MathExcpt
    |    |    |              \-SwegExcpt::SwegExcpt
    |    |    |-TMessages::WriteMessage
    |    |    |    |-TMessages::GetMsg
    |    |    |    |-TMessages::PrintALine
    |    |    |    |    \-TSeqFile::Write
    |    |    |    |         \-MFiles::Append
    |    |    |    \-sysdun
    |    |    |         |-TActWindow::GetNext
    |    |    |         |-TActWindow::~TActWindow
    |    |    |         \-ProgramStop::ProgramStop
    |    |    |              \-SwegExcpt::SwegExcpt
    |    |    |-TAddress::Spherical2Cartesian
    |    |    |    \-TMaster::DebugOn
    |    |    |-VertexIndex::operator=
    |    |    |-VerticeArray::operator[]
    |    |    |    \-VertexIndex::Value
    |    |    |-TTerrain::toPtr
    |    |    |    \-SwegExcpt::SwegExcpt
    |    |    |-operator-
    |    |    |    \-VertexIndex::VertexIndex
    |    |    |-VertexIndex::operator++
    |    |    |-dist
    |    |    |-operator+
    |    |    |    \-VertexIndex::VertexIndex
```

FIGURE 2.13-4. featlos Code Tree.

```
|    |    \-VertexIndex::operator+=
|    |         \-operator+
|    |-VerticeArray::operator[]
|    |-operator+
|    \-isZeroEquiv
|-TMemory::Index2Ptr
|-TMemory::AllocTemp
|    \-TMemory::DoAllocate
|         |-TMemory::GetBlockLength
|         |-CountMemOpns
|         |    \-TMaster::DebugOn
|         |-TMemory::Ptr2Index
|         |-ProgramStop::ProgramStop
|         \-TMemory::WriteSummary
|              |-TMemory::WordsUsed
|              |-TMemory::CalcWdsLeft
|              \-CountMemOpns
|-sorted_collection::sorted_collection
|-TAddress::GetAddresses
|    |-TAddress::GetCode
|    |    |-DVector::Getx
|    |    |-DVector::Gety
|    |    \-TMessages::WriteMessage
|    |-TAddress::InsertVertCodes
|    |    |-sorted_collection::insert_nodup
|    |    |    |-MTree::insert_nodup
|    |    |    |    |-MTree::insert_nodup
|    |    |    |    \-MTree::MTree
|    |    |    \-MTree::MTree
|    |    \-numerical
|    |-operator-
|    |    |-DVector::DVector
|    |    |-DVector::Getx
|    |    |-DVector::Gety
|    |    \-DVector::Getz
|    |-operator^
|    |    |-DVector::Getx
|    |    |-DVector::Gety
|    |    \-DVector::Getz
|    |-TAddress::GetCellRadius
|    |-dbg_sqrt
|    |-operator*
|    |    |-DVector::DVector
|    |    |-DVector::Getx
|    |    |-DVector::Gety
|    |    \-DVector::Getz
|    |-DVector::operator+=
|    |    |-DVector::Getx
```

FIGURE 2.13-4. featlos Code Tree. (Contd.)

```
|    |    |-DVector::Gety
|    |    \-DVector::Getz
|    \-operator+
|         |-DVector::DVector
|         |-DVector::Getx
|         |-DVector::Gety
|         \-DVector::Getz
|-sorted_collection::getfirst
|-sorted_collection::getnext
|    \-MTree::getnext
|-TAddress::GetShapeList
|    |-TAddrNode::DataPresent
|    |    \-TAddrNode::GetNode
|    |         |-TAddrData::GetCode
|    |         |-TAddrData::TAddrData
|    |         |-TAddrNode::TAddrNode
|    |         |    \-MTree::MTree
|    |         \-TAddrData::PutNodePtr
|    \-TAddrData::GetShapeList
|         \-TMemory::Index2Ptr
|-TMemory::Ptr2Index
|-TTable::SearchInt
|    \-TMemory::Ptr2Index
|-region
|    |-DVector::Getx
|    |-DVector::Gety
|    |-dbg_atan2
|    \-dist
|-crslwp
|    |-DVector::DVector
|    |-DVector::DVector
|    |-dbg_sqrt
|    |-crslwc
|    |    |-operator-
|    |    |-DVector::Putz
|    |    |-operator^
|    |    |-dbg_sqrt
|    |    |-operator+
|    |    \-operator*
|    |-TMemory::Index2Ptr
|    |-DVector::operator
|    |-crslwl
|    |    |-operator-
|    |    |-operator*
|    |    |    |-DVector::DVector
|    |    |    |-DVector::Gety
|    |    |    |-DVector::Getz
|    |    |    \-DVector::Getx
```

FIGURE 2.13-4. featlos Code Tree. (Contd.)

```
|    |    |-DVector::Getz
|    |    |-operator+
|    |    \-operator*
|    |-TMemory::Allocate
|    |    \-TMemory::DoAllocate
|    |-DVector::Getx
|    |-DVector::Gety
|    \-TMemory::Ptr2Index
|-DVector::DVector
|-sorted_collection::delete_collection
|    \-MTree::delete_tree
|         |-MTree::delete_tree
|         \-TMemory::Deallocate
|              |-TMemory::Deallocate
|              |    |-TMemory::DeallocFront
|              |    |    \-TMemory::GetBlockLength
|              |    |-TMemory::Index2Ptr
|              |    |-CountMemOpns
|              |    \-TMemory::RcylBlock
|              |         |-TMemory::Index2Ptr
|              |         \-TMemory::Ptr2Index
|              \-TMemory::Ptr2Index
\-TMemory::Deallocate
```

FIGURE 2.13-4. featlos Code Tree. (Contd.)

2.13.4 Assumptions and Limitations

- Distances are based upon a projection of a spherical earth onto a flat plane. The radius of the earth is set to 6371221.3 meters.
- The altitude of the surface of the earth is represented by a continuous, but not necessarily differentiable, function of x and y.
- SWEG does not model clutter, multipath, or diffraction, but it does simulate masking effects for sensing, communicating, and jamming.

2.13.5 Known Problems or Anomalies

None.